Санкт-Петербург, ул. Седова, 80, телефон (812) 449-0770, факс (812) 449-0771, E-mail: info@bilim.com

Управление трафиком с помощью правил iptables

Николай Малых

nmalykh@bilim.com

Достаточно часто возникает необходимость управления трафиком на граничных шлюзах, серверах или конечных станциях с использованием пороговых значений скорости. В ядре ОС Linux поддерживается механизм Token Bucket Filter (TBF)[1], который используется некоторыми дисциплинами управления очередями (в частности, sch_tbf) и условиями iptables. Для более наглядного представления о работе этого алгоритма вспомним задачку из школьной арифметики о бассейне с двумя трубами. Предположим, что у нас имеется сосуд объема V, из которого имеющаяся в нем жидкость вытекает с постоянной скоростью p. Поток жидкости, поступающей в бассейн, имеет переменную скорость. Очевидно, что при такой постановке задачи через некоторое время жидкость может наполнить сосуд и начать переливаться через край в том случае, если средняя скорость скорость притока жидкости превышает фиксированную скорость оттока. Ясно также, что продолжительность наполнения сосуда будет зависеть от определенное значение, которое зависит от скорости вытекания и объема сосуда от 3 параметров – фиксированных значений объема и скорости оттока, а также переменной скорости притока. Момент заполнения бассейна до краев может служить пороговым значением. Поскольку этот момент достаточно точно фиксируется, мы можем создавать правила управления потоком, которые определяют поведение системы до заполнения сосуда и после его наполнения. Если вернуться к управлению пакетами, то мы можем создавать правила, которые будут выполнять определенные действия до тех пор, “пока сосуд не наполнится” или наоборот, включать некий механизм (например, отбрасывание пакетов) при “наполнении сосуда”.

Как вы наверное помните, такие задачи в начальной школе решались достаточно легко. Не составляет большого труда решить их и программными средствами современных ОС. В широко распространенной системе фильтрации трафика netfilter/iptables существует несколько соответствий (match) работающих на основе этого алгоритма. К числу таких соответствий относятся достаточно хорошо известные limit, iplimit и недавно созданные hashlimit и tbf.

Соответствие limit

Условие, позволяющие ограничить частоту тех или иных событий, было реализовано самым первым в модуле limit и предназначалось прежде всего для ограничения частоты записи о событиях в журнальные файл системы.
Опция **-m limit** позволяет задать пороговое значение частоты выполнения условий, по достижении которого выполняется заданная правилом операция. Такая возможность весьма полезна для организации записи о событиях в системные журналы с помощью операций **LOG** (стр.) и **ULOG** (стр.). Вы можете делать запись в журнал не для каждого случая совпадения с заданными условиями, а лишь ограничиваться заданной в правиле iptables частотой таких событий. Это позволяет снизить размер журнальных файлов и сделать их более читаемыми.

Опция **-m limit**

может использоваться с параметрами

```
--limit <avg>
```

и

```
--limit-burst <burst>
```

Первый параметр задает среднюю частоту событий (скорость оттока) и указывается в формате **значение/суффикс**. Значение определяет число событий, а суффикс – единицу времени (**/s** или **/second** – секунда, **/m** или **/minute**– минута, **/h** или **/hour** – час, **/d** или **/day** – сутки). По умолчанию используется пороговая частота 3 пакета в час. Второй параметр определяет пик “разовой” доставки пакетов (объем сосуда). По умолчанию для параметра burst используется значение 5. Модуль работает следующим образом:

- условие считается выполненным, пока значение счетчика пакетов не превысит пика **limit-burst**;
- каждый пакет, соответствующий правилу, увеличивает значение счетчика на 1;
- по истечении каждого интервала 1/**limit** значение счетчика уменьшается на 1.

Вернемся к аналогии с задачей о бассейне с двумя трубами (в бассейн заданного объема вода втекает через одну трубу с переменной скоростью и вытекает с постоянной скоростью через другую трубу). Параметр **limit-burst** задает объем бассейна (количество помещающихся в него пакетов), а параметр **limit** определяет скорость оттока через выходную трубу. Пока в бассейне есть место, правило выполняется, а как только бассейн наполнится до краев, пакеты перестанут соответствовать правилу (польется через край). Очевидно, что в пустой бассейн может сразу поместиться **limit-burst** пакетов, а за счет вытекания через трубу число пакетов в бассейне уменьшается на **limit** в единицу времени. Следовательно за это время в бассейн можно поместить до **limit** новых пакетов. Если пакеты не приходят, бассейн постепенно опустошается и может принять в себя больший объем.

Очевидно, что срабатывание правил с этим условием имеет определенный гистерезис по частоте доставки пакетов, т. е. частота доставки, при которой условие “включается”, будет отличаться от частоты, при которой условие будет “выключено”. Рассмотрим работу модуля на примере правила

```
iptables -A FORWARD -m limit -j LOG
```

с принятыми по умолчанию значениями параметров. Информация о первых пяти пакетах, переданных правилу, будет записана в журнальный файл, поскольку значение счетчика не достигло пика. После этого, в течение 20 минут (1/3 часа) записи производиться не будут, независимо от частоты поступления пакетов. В конце этого интервала значение счетчика уменьшится на 1 и в течение следующего 20-минутного интервала в журнальный файл может быть записан еще один пакет (т. е., будет записываться в информация о первом пакете в течение каждой 20-минутки). Счетчик пикового значения уменьшается на 1 каждые

[1] Его часто называют также Leaky Bucket, поскольку его работу очень хорошо иллюстрирует ситуация с вливанием жидкости в сосуд, из которого обеспечивается отток с постоянной скоростью (дырявое ведро).

20 (60/30) минут при отсутствии пакетов, поэтому, если в течение 100 минут (60/3 * 5) не поступит ни одного пакета, счетчик пикового значения достигнет нуля и процесс начнется заново.

Отметим, что при выборе частоты 1 пакет в сутки, значение пика не может быть меньше 3 (это связано с ограничениями используемых для счетчиков типов целых чисел).

Вы можете использовать этот модуль для защиты от атак на службы (DoS).

Защита от атак SYN-flood:

```
iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
```

Защита от хитроумных сканеров портов:

```
iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
```

Защита от Ping of death:

```
iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

Для использования этого условия в ядре должна быть включена поддержка опции **limit match support** Если для опции выбрано значение M, потребуется загрузка модуля ядра **ipt_limit**.

Поскольку условие не поддерживает инверсии, использование правил с таким условием в реальных фильтрах существенно осложняется. Например, для приведенной выше защиты от атак SYN-flood потребуется дополнительное правило, как показано ниже

```
iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
iptables -A FORWARD -p tcp --syn -m limit -j DROP
```

Однако, в реальных условиях все может оказаться сложнее и такой простой конструкции окажется недостаточно. Логика построения цепочек iptables не допускает переходов (типа go to в программах) и позволяет лишь передавать пакеты в пользовательские цепочки (по типу процедур языка Паскаль), откуда они возвращаются в точку вызова, если обработка пакета не завершается в пользовательской цепочке. В приведенном примере нам остается лишь отбросить (строка 2) все пакеты, которые не соответствуют правилу строки 1, а в реальных условиях может потребоваться значительно более изощренная фильтрация.

Существенное расширение возможностей использования условия limit было бы обеспечено простой возможностью инверсии правила в целом, однако такая функция не поддерживается ни данным модулем, ни его наследником hashlimit. Однако в последнем модуле этой серии tbf возможность инверсии правила в целом поддерживается, что существенно повышает уровень гибкости.

Соответствие iplimit

Условие iplimit позволяет ограничить число одновременных (параллельных) соединений TCP для каждого хоста независимо. Например, вы можете таким путем ограничить число соединений с Web-сервером для каждого клиента независимо:

```
iptables -A INPUT -p tcp --syn --dport http -m iplimit --iplimit-above 4 -j REJECT
```

Модуль также позволяет ограничивать число соединений для всех хостов подсети, заданной маской:

```
iptables -A INPUT -p tcp --syn --dport http -m iplimit --iplimit-mask 8 --iplimit-above 4 -j
REJECT
```

Модуль использует два параметра. Первый параметр

```
[!] --iplimit-above n
```

определяет пороговое значение числа соединений, а второй (необязательный) параметр определяет маску (размер блока) адресов для которых будет использоваться общий счетчик числа пакетов

```
--iplimit-mask n
```

Значение n задает размер маски адреса в битах. Например, использование единого счетчика пакетов для всех хостов сети класса B обеспечивается заданием маски 16.

Условие iplimit поддерживает инверсию для первого параметра, что позволяет создавать гибкие наборы правил. Например, вы можете использовать операцию ACCEPT до тех пор, пока количество параллельных соединений не превышает определенное значение, а в следующих строках задать правило, определяющее судьбу пакетов, которые не соответствуют данному правилу. Это может быть не только отбрасывание пакетов, но и запись о них в системный журнал, передача пакетов в пользовательские цепочки и лр.

Соответствие hashlimit

Модуль hashlimit является прямым наследником limit и основным (весьма кардинальным) отличием является поддержка хеш-таблиц, которые позволяют поддерживать и проверять условия соответствия не для всех пакетов сразу (независимо от адресов и портов), а с учетом выбранной пользователем комбинации параметров, включающей адреса и номер порта для отправителя и получателя. Это существенно расширяет возможности управления трафиком, поскольку выполнение условий проверяется независимо для каждого набора параметров в соответствии с выбором пользователя.
Основная идея модуля заключается в создании хэш-таблицы для каждого адреса получателя и поддержка отдельных счетчиков пакетов и байтов для каждой записи. Таким способом можно, например, вводить ограничения на скорость и число попыток организации соединения с каждым адресом, как это делает **limit** для всех адресов сразу.

Модуль поддерживает несколько параметров:

```
--hashlimit <avg>
```

задает среднее значение потока трафика. Параметр представляет собой целое число, определяющее максимальное количество пакетов, и суффикс, который определяет единицу времени. В качестве суффикса могут использоваться значения **/second**, **/minute**, **/hour**, **/day**. По умолчанию устанавливается число пакетов в секунду.

```
--hashlimit-burst <burst>
```

задает порог, идентичный параметру **limit-burst** для операции **limit** (По умолчанию значение параметра равно 5.

```
--hashlimit-mode <режим>
```

задает режим хеширования – по адресам, портам или их комбинациям. Параметр может содержать разделенный запятыми список значений **dstip**, **dstport**, **srcip**, **srcport**. Эта опция является обязательной в команде.

`--hashlimit-name foo`
задает имя для файла **/proc/net/ipt_hashlimit/foo**. Данная опция является обязательной. Имя файла может выбираться произвольно.

`--hashlimit-htable-size <num>`
задает число элементов (bucket) в хэш-таблице;

`--hashlimit-htable-max <num>`
задает максимальное количество записей в хэше;

`--hashlimit-htable-expire <num>`
задает время (в миллисекундах) жизни записи в хэш-таблице. По умолчанию время жизни составляет 10000 (10 секунд).

`--hashlimit-htable-gcinterval <num>`
задает интервал "сборки мусора" в хэш-таблице. По умолчанию интервал сборки составляет 1000 мсек (1 секунда).

Например, правило
```
iptables -A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -m hashlimit --hashlimit
1/hour --hashlimit-burst 2 --hashlimit-mode srcip --hashlimit-name SSH --hashlimit-htable-
expire 360000 -j ACCEPT
```
позволяет ограничить частоту попыток соединения с портом SSH независимо для каждого источника таких пакетов.
При использовании условия **hashlimit** я дереве каталогов **/proc/net/ipt_hashlimit** создается файл с именем, заданным параметром – **hashlimit-name** (в нашем случае **/proc/net/ipt_hashlimit/SSH**), в котором для каждого отправителя пакетов создается запись вида
```
50 213.96.92.236:0->0.0.0.0:0 23040000 23040000 11520000
```
Первый параметр показывает число секунд, в течении которого запись будет оставаться в таблице, если с этого адреса не поступит новых пакетов, соответствующих остальной части правила. При поступлении нового пакета для этого поля устанавливается значение, задаваемое параметром **–hashlimit-htable-expire** (а приведенном примере 360000=360 сек = 6 мин). Следующий параметр содержит значения адресов и номеров портов для отправителя и получателя с учетом выбранного пользователем режима хеширования. Если какой-либо из перечисленных параметров не используется при хеширование соответствующее поле таблицы имеет значение 0. Третий параметр (23040000 в приведенном примере) содержит текущее значение счетчика пакетов (не удивляйтесь большому значению, счетчик использует коэффициент), четвертый параметр пропорционален значению burst (с тем же коэффициентом), а последний параметр определяет средний интервал "оттока", т. е. обратно пропорционален значению параметра –hashlimit (в большинстве систем он будет равен среднему интервалу между пакетами в секундах, умноженному на 32000).
В связи с использованием хеш-таблицы модуль поддерживает еще три дополнительных параметра, управляющих выделением памяти для хеширования. Параметр **–hashlimit-htable-size** определяет размер памяти, выделяемой при создании хеш-таблицы (число записей), а **–hashlimit-htable-max** задает максимальное число записей в таблице. Эти параметры являются необязательными и при их отсутствии соответствующие значения вычисляются с учетом параметров системы. Третий параметр – **hashlimit-htable-gcinterval** определяет период "сборки мусора", т. е. Удаления из таблицы записей, для которых истекло время жизни.
Условие hashlimit существенно расширяет возможности фильтрации пакетов с использованием пороговых значений, но обладает тем же недостатком, что и limit – отсутствием возможности инверсии правила в целом.
Кроме того, исходные коды hashlimit в составе ядра 2.6.10 и iptables 1.3.0.r1 содержат ряд ошибок, которые просто не позволяют использовать модуль. Файлы с корректными исходными кодами можно загрузить с сайтов, используя приведенные ниже ссылки:
http://bugme.osdl.org/show_bug.cgi?id=4105 (модуль для ядра)
https://bugzilla.netfilter.org/bugzilla/show_bug.cgi?id=280 (модуль iptables)

Соответствие tbf

Модули для поддержки условия tbf были разработаны автором в конце 2004 года и пока не включены в распространяемые официально коды ядра и iptables. Вы можете загрузить файлы с сайта автора, воспользовавшись приведенной ниже ссылкой
http://www.nmalykh.org/work/tbf.tar.gz.
Этот модуль является прямым наследником limit и hashlimit, выполняя функции того и другого, но в отличие от своих предшественников поддерживает возможность инверсии правила в целом, что существенно повышает уровень гибкости и расширяет возможность создания эффективных цепочек iptables для использования в реальных условиях на граничных шлюзах, серверах и пользовательских станциях, работающих под управлением ОС Linux с ядром 2.6.x.
Модуль поддерживает несколько параметров:

`--tbf <avg>`
задает среднее значение потока трафика. Параметр представляет собой целое число, определяющее максимальное количество пакетов, и суффикс, который определяет единицу времени. В качестве суффикса могут использоваться значения **/second**, **/minute**, **/hour**, **/day**. По умолчанию устанавливается число пакетов в секунду.

`--tbf-deep <burst>`
задает порог, идентичный параметру **limit-burst** для операции **limit** (По умолчанию значение параметра равно 3.

`--tbf-mode <mode>`
задает режим хеширования – по адресам, портам или их комбинациям. Параметр может содержать разделенный запятыми список значений **dstip**, **dstport**, **srcip**, **srcport**, а также значения **all** и **nothing**. Значение **all** эквивалентно заданию комбинации из всех 4 параметров **dstip**, **dstport**, **srcip**, **srcport** и введено просто для сокращения записи. Значение **nothing** переводит модуль в вырожденный режим, когда хеш-таблица содержит единственную запись для всех комбинаций адресов и номеров портов отправителя и получателя. В этом режиме модуль работает аналогично модулю limit, отличаясь от последнего лишь созданием файла в каталоге **/proc/net/ipt_tbf** и поддержкой инверсии для правила в целом. Эта опция является обязательной в команде.

`--tbf-name <filename>`
задает имя для файла **/proc/net/ipt_tbf/foo**. Данная опция является обязательной. Имя файла может выбираться произвольно.

`--tbf-htable-size <num>`
задает число элементов (bucket) в хэш-таблице;

`--tbf-htable-max <num>`
задает максимальное количество записей в хэше;

`--tbf-htable-expire <num>`

задает время (в миллисекундах) жизни записи в хэш-таблице. По умолчанию время жизни составляет 10000 (10 секунд).

`--tbf-htable-gcinterval <num>`

задает интервал “сборки мусора” в хэш-таблице. По умолчанию интервал сборки составляет 1000 мсек (1 секунда).

Перечислим кратко основные отличия модуля tbf от его предшественников

1. Название параметра, определяющего “объем сосуда” изменено с -burst на -deep, что на мой взгляд более точно отражает физический смысл этого параметра.
2. Изменено принятое по умолчанию значение параметра **deep**, что позволило устанавливать значение средней частоты 1 пакет/сутки без возникновения конфликтов, присущих модулям limit и hashlimit.
3. Расширен набор опций режима хеширования за счет поддержки значений **all** и **nothing**.
4. Модуль заменяет собой модули limit и hashlimit, упрощая создание больших таблиц фильтрации пакетов.
5. Обеспечивается возможность инверсии на уровне правила iptables в целом (соответствует – не соответствует) что с учетом специфики создания правил iptables существенно расширяет возможности фильтрации.

Рассмотрим преимущества, обеспечиваемые поддержкой инверсии на примере фильтрации спама на граничном шлюзе. При использовании модуля hashlimit мы может задать правило, определяющее пороговое значение числа попыток организации соединения с портом SMTP для каждого хоста Internet независимо. Фрагмент такого правил приведен ниже

```
-m state --state NEW -m hashlimit --hashlimit 1/s --hashlimit-burst  --hashlimit-mode srcip
--hashlimit-name SMTP
```

В правиле пока не указана операция (TARGET) поскольку мы сейчас рассмотрим возможные варианты. Приведенному правилу пакеты организации новых соединений SMTP будут соответствовать до тех пор, пока частота из прибытия с данного адреса не превысит заданный порог 1 пакет в секунду. Логика и учет специфики последовательной обработки правил в iptables и отсутствием возможности переходов диктует в качестве операции для этого правила единственный вариант - ACCEPT, не позволяя организовать дополнительных проверок для таких пакетов. Формально мы можем указать в качестве операции пользовательскую цепочку, содержащую дополнительные проверки и отбрасывание пакетов по из результатам, но по завершении обработки правил пользовательской цепочки пакет попадает в правило, которое следует за правилом, содержащим приведенный выше фрагмент. Не снижая уровня общности мы можем предположить два варианта судьбы пакетов организации новых соединений SMTP после их обработки в следующем правиле – восприятие пакета для дальнейшей обработки или его отбрасывание. Рассмотрим оба варианта

1. **ACCEPT**. Мы принимаем пакеты, которые возвращены из пользовательской цепочки, указанно в качестве TARGET для строки hashlimit. Но эта же строка обеспечит восприятие пакетов, которые не соответствуют условию hashlimit (слишком частые попытки организации соединений с одного адреса). В результате мы отфильтруем часть спама в пользовательской цепочке, но пропустим все остальное в правиле, следующем за строкой hashlimit.

2. **DROP**. Мы отбрасываем все пакеты, которые связаны со слишком частыми попытками организации соединений с одного адреса (пакеты не соответствуют условию hashlimit), но вместе с тем мы отбросим и все пакеты, возвращенные из пользовательской цепочки, которая указана в качестве операции hashlimit. Следовательно, эта пользовательская цепочка должна не отбрасывать пакеты с используемых для рассылки спама хостов, а наоборот, принимать пакеты с легитимных адресов. Логически это не приводит к серьезным ограничениям на создание пользовательской цепочки, но практика показывает, что перечислить блокируемые адреса проще, нежели легитимные (правил получается меньше).

Если же мы можем применять и инверсию типа

```
-m state --state NEW -m tbf ! -- tbf 1/s --tbf-burst --tbf-mode srcip --tbf-name SMTP
```

это позволяет сразу отбросить слишком частые попытки организации соединений SMTP с одного адреса, а в последующих строках произвести дополнительные проверки (например, на принадлежность к блокам адресов, используемых для коммутируемого доступа.

Модуль был протестирован на внутренних хостах сети, а с января 2005 года используется на реальных шлюзах, связывающих локальную сеть компании BiLiM Systems (www.bilim.com) с Internet, и публичных серверах, поддерживаемых нашей компанией.

Автор модуля будет признателен всем, кто пожелает протестировать его в реальных условиях и выскажет свое мнение или совет.